PowerRush: A Linear Simulator for Power Grid* †

Jianlei Yang Zuowei Li Yici Cai Qiang Zhou
Department of Computer Science and Technology
Tsinghua University, Beijing, China
yjl09@mails.tsinghua.edu.cn

ABSTRACT

As the increasing size of power grids, IR drop analysis has become more computationally challenging both in runtime and memory consumption. In this paper, we propose a linear complexity simulator named **PowerRush**, which consists of an efficient SPICE Parser, a robust circuit Builder and a linear solver. The proposed solver is a pure algebraic method which can provide an optimal convergence without geometric information. It is implemented by Algebraic Multigrid Preconditioned Conjugate Gradient method, in which an aggregation based algebraic multigrid with K-Cycle acceleration is adopted as a preconditioner to improve the robustness of conjugate gradient iterative method. In multigrid scheme, double pairwise aggregation technique is applied to the matrix graph in coarsening procedure to ensure low setup cost and memory requirement. Further, a K-Cycle multigrid scheme is adopted to provide Krylov subspace acceleration at each level to guarantee optimal or near optimal convergence. Experimental results on real power grids have shown that **PowerRush** has a linear complexity in runtime cost and memory consumption. The DC analysis of a 60 Million nodes power grid can be solved by **PowerRush** for $0.01mV$ accuracy in 170 seconds with 21.89GB memory used.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Simulation*

Keywords

PowerRush, Power Grid, Algebraic Multigrid, Aggregation, K-Cycle

1. INTRODUCTION

The design and analysis of extremely large scale power grids is a very computationally challenging task for VLSI design. Many contributions have been developed, such as Krylov-subspace method [1], Hierarchical and Macro-modeling methods [2], Random Walk [3], Domain Decomposition [4], SPAI [5], $\mathcal{H}$-Matrix [6]. Geometric based multigrid like technique was exploited to efficient power grid analysis [7]. And in order to further handle irregular power grids, algebraic multigrid (AMG) based techniques [8][9] were also developed.

For algebraic multigrid approaches, there exists a trade-off between setup cost and efficiency of error components reduction. The grid reduction methods in some AMG-like multigrid approaches [8] are also somewhat geometrically based which may degrade the efficiency due to the irregularity. A pure algebraic multigrid method [10] was proposed to improve the efficiency whose grid reduction was based on matrix graph. But its denser prolongation matrices increase setup costs and memory requirements. To overcome the bottleneck of limited convergence by controlling of the number of coarsening levels, smoothed aggregation method [11] is developed recently which is robust and efficient over a wide variety of problems. Aggregation based algebraic multigrid is also introduced to power grid analysis [12]. In [13], Fourier analysis has been explored for several aggregation based two-grid schemes for a model anisotropic problem. But in practice it is too difficult to hope for optimal order convergence with the V-Cycle or even with the standard W-Cycle.

Aiming to obtain an optimal convergence with low setup cost, in this work, we propose a pure algebraic multigrid solver for irregular power grid simulation. The contributions of our work are: **(i)** In our multigrid approach, double pairwise aggregation strategy [14] is introduced to reduce the grids effectively to small size by multilevel matrices representation which requires no explicit knowledge of the problem geometry. Further, the restriction and prolongation matrices are easy to construct with very low memory consumption. **(ii)** In order to increase the robustness of standard multigrid approaches, AMG is adopted as an implicit preconditioner for conjugate gradient iteration routine instead of stand-alone solver [15]. In this scheme, the main iteration routines smooth out high frequency errors rapidly and AMG uses a projection of the fine-grid problem on a coarser grid to remove the low frequency error components. **(iii)** Compared with previous aggregation based multigrid methods [12] in power grid analysis, the scalability is enhanced by K-Cycle multigrid scheme to provide Krylov subspace acceleration at each level. Related works have shown that K-Cycle multigrid can provide optimal or near optimal convergence under mild assumptions on the two-grid scheme [16].

The reminder of this paper is organized as follows. Section 2 gives a brief background of power grid analysis. In Section 3 we present a simulation flow of our simulator **PowerRush**. Following that, AMG-PCG solver and its practical implementation use is discussed in Section 4. Experimental results are provided and numerically illustrated in Section 5. Concluding remarks are given in Section 6.

*This work is supported by National Natural Science Foundation of China (NSFC) No.60976035.

†This paper is invited by Special Session of *2011 TAU Power Grid Simulation Contest* in ICCAD 2011.

2. BACKGROUND

2.1 Power Grid Analysis

In this section, we give the basic modeling and analysis techniques for efficient and accurate analysis. Typically, power grid on die is designed from top-level metal layer, which is connected to the package, down through inter-layer vias and finally to the active devices. For DC simulation, power grid can be modeled as linear resistive network system. By using Modified Nodal Analysis (MNA) method, a n-node circuit network can be formulated as a linear system equations [1]. After reformulation by *Norton's Law* this system equations can result in a symmetric, positive definite problem whose system matrix is a non-singular $\mathcal{M}$-matrix [8]:

$$Gu = I \tag{1}$$

where $u \in \mathbb{R}^{n\times 1}$ is an unknown vector of node voltages, $G \in \mathbb{R}^{n\times n}$ is the conductance matrix, $I \in \mathbb{R}^{n\times 1}$ is a vector of node current sources. The diagonal entries of matrix G are defined by $g_{ii} = \sum_{j\in N_i} |g_{ij}|$, where $N_i = \{j \,|g_{ij} \neq 0\}$ is the set of neighbors of node i, g_{ij} defines the conductance between the two neighboring nodes i and j, thus $g_{ij} = g_{ji}$ which results in the G matrix being symmetric.

As the VLSI technology scaling associated with significantly increasing device numbers in a die, the number of nodes in the power grid may easily exceed many millions. The most accurate and stable methods for solving such huge linear systems are sparse direct solvers such as *SuperLU* and *Cholmod*, but both of them are time expensive and memory inefficient. Another state of art approach is iterative methods especially preconditioned iterative methods which can be used to solve such linear systems with memory efficiently. However, preconditioned iterative methods are not stable for many cases because of either expensive cost or unsatisfactory performance of their preconditioners.

2.2 Algebraic Multigrid Method

Multigrid methods [17] are called scalable and optimal which are expected to solve a linear system within linear complexity. Multigrid methods achieve optimality through the effect of a smoother and a coarse grid correction. In multigrid scheme, the smoother is fixed and generally based on a simple iterative relaxation method. The coarse grid correction involves transferring information to a coarse grid through restriction and computing an approximate solution to the residual equation on a coarser grid, which is to say, solving a linear system of smaller size. This solution is then transferred back to the original grid by means of an appropriate interpolation which is also called prolongation. In the classical multigrid setting, smoothing reduces high frequency error, whereas coarse grid correction eliminates low frequency error.

Due to the irregularity of real power grid designs [18], AMG methods have been well developed in power grid simulation area. AMG method [19] determines coarse grids, inter-grid transfer operators, and coarse-grid equations based solely on the matrix entries. Although the classical AMG methods work remarkably well for a wide variety of problems, some of the assumptions make in its derivation limit its applicability. The smoothed aggregation method [11] is a highly successfully AMG method that is robust and efficient over a wide variety of problems. The most interesting aspect of aggregation based AMG is its approach to define interpolation. The aggregation algorithm first partitions the grid by aggregation grid points into small disjoint sets and then builds a preliminary interpolation operator to coarsen power grids by system matrix level. Thus, the prolongation matrices with at most one nonzero entry per row are much sparser than the ones obtained by the classical AMG approach. Numeric analysis has shown that for two-dimensional anisotropic model problem aggregation based two-grid methods [13] may have optimal order convergence properties.

3. SIMULATION FLOW

This work presents a friendly fast simulator with linear complexity for power grid network. The simulator consists of a smart SPICE Parser, a robust circuit Builder and a linear complexity Solver. The detailed simulation flow of our simulator **PowerRush** is shown in Figure 1.

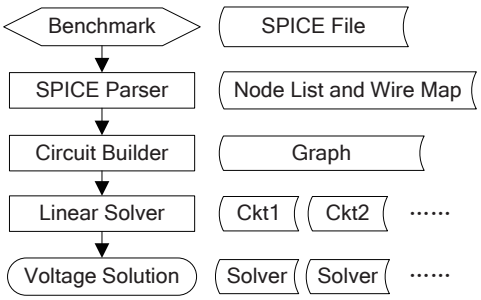


Figure 1: Simulation Flow of PowerRush.

In parser procedure, it hashes each node name to index. The **hash** function we used in our approach ensures the average search length of each node is about 1.16 and the maximum search length is no more than 6. As we have known that the complexity of hash method to find an element in a table is expected to $\mathcal{O}(1)$. So, the total complexity of our parser is expected to $\mathcal{O}(N)$ which N is the size of grid nodes. After parser process for the SPICE file, the circuit is stored as nodes list and wires map which are linked to present their topologies.

The most important part of building circuit is to create a graph to handle all wires map and nodes list. The most frequently appearance in this procedure is short path between two nodes which must be merged as one equivalent node. The number of unknowns can be reduced by nodes merging for short path, such as vias with very small resistance value or zero resistance value. The **disjoint** data structure is used to find and union nodes sets whose complexity is $\mathcal{O}(logN)$. Further, it is noticeable that there exist several separated nets for each benchmark in real power grid designs [18]. So our builder identifies all separated nets by **DFS** (Depth-First Search) algorithm whose complexity is $\mathcal{O}(N)$ and then solve each net independently. Thus, the total solving cost can be added linearly by each net.

After building the graph of power grid, we build the conductance matrix G by MNA method and the corresponding vector of current sources. Then the linear system is solved by AMG-PCG method which stands for **A**lgebraic **M**ulti**G**rid **P**reconditioned **C**onjugate **G**radient method. Among AMG-PCG solver, aggregation based algebraic multi-

grid with K-Cycle acceleration is adopted as a preconditioner to improve the robustness of conjugate gradient iterative method [20].

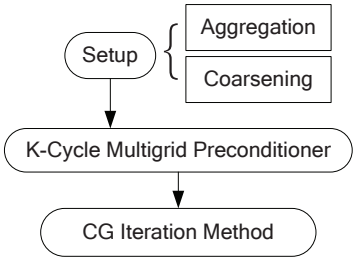


Figure 2: AMG-PCG Solver.

As shown in Figure 2, the application of AMG-PCG to a given problem is a three part process. The first part, which is a fully automatic setup phase, consists of recursively choosing the coarser levels and defining the transfer and coarse-grid operators. The second part, which is the preconditioning phase, just uses the resulting components in order to perform acceleration recursively on all levels by the use of K-Cycle scheme. Thirdly, aggregation based algebraic multigrid with K-Cycle accelerating is adopted as an implicit preconditioner for CG iteration method to solve the power grid. The details of this linear solver are demonstrated in Section 4.

4. AMG-PCG SOLVER

In this work, we use aggregation based algebraic multigrid method with K-Cycle acceleration to improve the convergence. This procedure is fully algebraic, that is, it works with the information present in the system matrix only.

4.1 Aggregation Coarsening

As demonstrated in Figure 2, the aggregation scheme contains two steps. First, we need to partition the unknowns into disjoint subsets to generate the prolongation matrices. Then, the prolongation matrices are used to formulate the coarse matrix.

As described in [14], we focus on schemes that use coarsening by aggregation where the strongest connection is favored in forming pairs. With aggregation scheme, the fine grid unknowns are grouped into disjoint subsets, and each such subset is associated to a unique coarse level unknown. Prolongation from coarse level to fine level is a vector defined on the coarse variable set by assigning the value at a given coarse variable to all fine grid variables associated to it. The prolongation operator P is a Boolean matrix with exactly one nonzero entry in each row and piecewise constant in each column. The restriction operator R is chosen to be the transpose of the prolongation. The coarse grid matrices are then cheap to compute and generally as sparse as the original fine grid matrix. As we have denoted the conductance matrix G as fine grid matrix A_f which is symmetric and positive definite, the coarse grid matrix is computed from the Galerkin formula $A_c = RA_fP = P^TA_fP$ which implies that A_c is cheap to construct using

$$(A_c)_{ij} = \sum_{k \in G_i} \sum_{l \in G_j} a_{kl} \tag{2}$$

That is, the entries in the coarse grid matrix are obtained by summing the entries in A_f that connect the different aggregates.

The above simple pairwise aggregation coarsening is still relatively slow, which cannot guarantee an optimal performance of multilevel methods. This work adopts the double pairwise aggregation algorithm which begins by forming pairs in the scaled problem matrix [20]. This fast coarsening technique is implemented in our simulator by repeating the simple pairwise aggregation process, defining aggregates by forming pairs of pairs.

Furthermore, preliminary numerical results indicate that aggregation based multigrid methods may then indeed exhibit convergence that is independent or near-independent of the number of levels [13].

4.2 Multilevel Preconditioning

Although AMG strategy has been tried to capture all relevant influences by accurate coarsening and interpolation, its interpolation will hardly ever be optimal. Multilevel error reduction technique can smooth the majority of the error components very quickly but inefficiency for just a few exceptional error components. Thus, its total convergence property is degraded by this non-ideal phenomenon. In order to improve the robustness of our approaches, AMG is used as an implicit preconditioner for conjugate gradients method instead of stand-alone solver [15].

The AMG preconditioner at each level on a given vector is computed according to the following algorithm [20]:

Algorithm 1: AMG as preconditioner at level k, $AMGprecond\,(r_k, k)$

Input: The residual r_k of level k
Output: The preconditioned vector z_k

1. Pre-smoothing: relax the input residual r_k to v_k by using smoother of several iterations on $A_k v_k = r_k$;
2. Compute a new residual $\tilde{r}_k = r_k - A_k v_k$ and restrict it to coarser grid by $r_{k-1} = P_k^T \tilde{r}_k$;
3. Compute an (approximate) solution y_{k-1} on coarser grid by $A_{k-1}y_{k-1} = r_{k-1}$ which is recursively obtained by $y_{k-1} = AMGprecond\,(r_{k-1}, k-1)$ until the coarsest level with $k = 1$;
4. Interpolate coarse grid correction by $y_k = P_k y_{k-1}$ and compute new residual $\bar{r}_k = \tilde{r}_k - A_k y_k$;
5. Post-smoothing: relax the new residual $\bar{r}_k$ to w_k by using smoother of several iterations on $A_k w_k = \bar{r}_k$;
6. Obtain preconditioned vector $z_k = v_k + y_k + w_k$;

The multigrid preconditioning is called by the main iteration routine at the top level, and it recursively calls itself in step 3 with a smaller index until the coarsest level. We stop the coarsening when the coarse grid matrix has 400 rows, allowing fast direct solver with sparse Cholesky factorization. For $k > 1$, the AMG preconditioner at level k computes y_{k-1} approximately by using AMG preconditioner at level $k-1$. The way this is done defines the cycling strategy which will be detailed discussed in next subsection.

4.3 K-Cycle Multigrid

This paper adopts a K-Cycle multigrid [16], with which Krylov subspace acceleration is applied at every level to enhance robustness and scalability.

As shown in Figure 3, it illustrates Krylov subspace ac-

Table 1: Simulation results of IBM Power Grid Benchmarks

Benchmark	Grid Size		Simulation Time (second)			Peak Memory (MB)	Max Error (mV)	Average Error (mV)
	Before†	After*	Parse	Build	Solve			
ibmpg3	851581	286299	2.79	0.52	2.48	230.03	0.07	0.0124
ibmpg4	953580	610103	2.89	0.80	3.31	309.82	0.23	0.0290
ibmpg5	1079307	540497	2.94	0.83	2.64	301.88	0.06	0.0117
ibmpg6	1670491	834633	4.82	1.38	6.67	466.20	0.09	0.0122
ibmpgnew1	1461038	531777	4.27	1.27	5.08	384.31	0.07	0.0124
ibmpgnew2	1461038	531777	4.82	1.00	5.06	397.84	0.07	0.0124

† *Before* means the original grid size.
* *After* means the grid size after merging the short path as equivalent node.

celeration recursively on each level. The Krylov-subspace smoothing is performed at the end of every recursive cycle on each level to reduce the residual, in which it is called inner iteration.

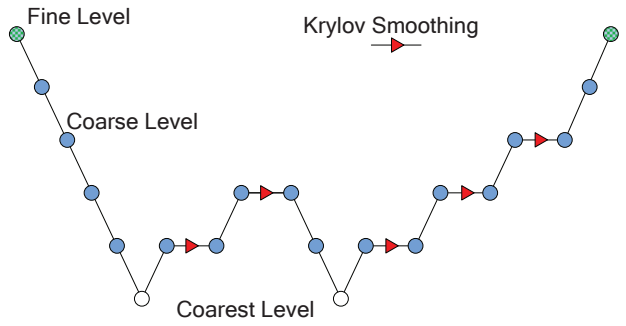


Figure 3: K-Cycle Multigrid.

In this scheme, the residual on coarse grid system is smoothed by several steps of a Krylov subspace iterative method. This approach is followed recursively until the coarsest level where an exact solver is performed. In practice, the K-Cycle with only 2 inner iterations at each level is observed to be optimal.

Theoretical analysis [13][16] have been shown that aggregation based multigrid with K-Cycle can achieve a guaranteed convergence which is independent or near-independent of the number of levels. K-Cycle multigrid appears more robust than V-Cycle or even standard W-Cycle. This enhanced robustness is obtained nearly for free since the K-Cycle has roughly the same computational complexity as the V-Cycle or W-Cycle.

5. EXPERIMENTAL RESULTS

All algorithms in **PowerRush** are implemented by C/C++ language with single thread. The simulation platform is a Red Hat Enterprise Linux Advanced Server with 2 Quad-Core Intel Xeon E5506 CPU@2.13GHz and 24GB RAM on HP ProLiant DL380 G6 Server.

Many efforts of detail implementations are taken into the parser step and the circuit build step of PowerRush to guarantee a stable simulator for more general power grid designs. Each benchmark is carefully checked in parser procedure to merge the short path as equivalent node then its original grid size is reduced to a smaller one. Also the resistors with extremely small resistance value are neglected, and bad vias between each two neighbor layers are handled adaptively.

The runtime in simulation flow is measured by system clock counter. The peak memory usage is measured by a friendly tool **Memtime** whose memory usage is fetched from a PID information of /proc/[PID]/stat. In our simulator, the tolerance of linear solver is set as 10^{-6} on the relative residual norm to guarantee that all voltage solution error is less than $0.01mV$.

5.1 Simulation on Industrial Power Grid

Firstly we carried our various experiments on industrial power grids which are drawn from real designs [18] to validate the promising performance of the proposed simulator **PowerRush**. As shown in Table 1, the second column is the original grid size of each benchmark, and the second column is equivalent grid size. The max voltage errors of most benchmarks are less than $0.1mV$ and the average voltage errors of most benchmarks are less than $0.02mV$. The runtime and memory usage are relative small and increase slowly with the grid size increasing. Experimental results have shown that **PowerRush** is effective and robust with high accuracy for real power grids analysis.

5.2 Simulation on More Larger Power Grid

Aiming to present the promising linear complexity of **PowerRush**, a serious of larger scale power grid benchmarks are generated by our power grid planner, in which the grid size is from **5 Million** to **60 Million** as shown in Table 2. Among them, there exists only one single net in each benchmark, which means that there are no separated nets in a same benchmark. That is to say, it cannot take the advantage of divide and conquer strategy in circuit building process as described in Section 3.

In Table 2, it illustrates the behavior of our proposed simulator **PowerRush** on our generated power grids. The complexity character of each step in **PowerRush** is shown in Figure 4. The parsing time, building time, solving time and peak memory usage are all linear complexity with the increasing size of power grids. We also compare the performance of **PowerRush** and Cholmod solver [21] which is included in SuiteSparse packages. The last two columns list the solving time and memory usage of Cholmod which not include the parser and builder cost. As shown in Table 2, AMG-PCG solver has obtained many speedups and very low memory consumptions. For the largest two benchmarks, Cholmod can not handle such huge scale grids because of much more memory required.

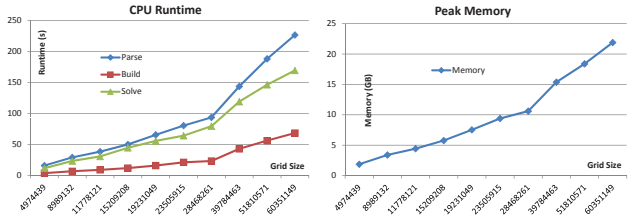
5.3 Grid Reduction Efficiency

Also in Table 4, it presents the grid reduction efficiency of double pairwise aggregation strategy. For all benchmarks,

Table 2: Simulation results of THU Power Grid Benchmarks

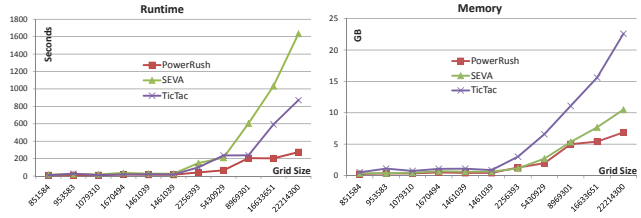
			PowerRush					Cholmod Solver	
	Grid Size		Simulation Time (second)				Peak Memory	Time (second)	Memory
Benchmark	Before†	After*	Parse	Build	Iterations	Solve			
thupg1	4974439	3261850	15.78	3.53	6	11.41	1.86GB	125.94	1.93GB
thupg2	8989132	6014504	28.89	6.57	6	23.24	3.39GB	305.39	3.69GB
thupg3	11778121	7856200	38.25	8.88	7	30.52	4.43GB	438.59	4.90GB
thupg4	15209208	10259771	49.86	11.73	6	44.48	5.76GB	696.96	6.58GB
thupg5	19231049	13627506	65.46	15.78	7	55.59	7.52GB	1203.88	8.66GB
thupg6	23505915	17216192	80.34	21.05	6	64.04	9.40GB	1867.98	11.30GB
thupg7	28468261	18690184	93.63	23.13	7	79.42	10.60GB	1823.61	12.43GB
thupg8	39784463	27661623	143.62	42.79	6	118.98	15.37GB	3011.56	18.54GB
thupg9	51810571	31518068	188.16	56.07	6	146.34	18.38GB	-	-
thupg10	60351149	38415188	226.72	68.24	6	169.42	21.89GB	-	-

† *Before* means the original grid size.
* *After* means the grid size after merging the short path as equivalent node.



(a) Runtime (b) Memory

Figure 4: Runtime and Memory Complexity of PowerRush.



(a) Runtime (b) Memory

Figure 5: Runtime and Memory Chart of Top 3 Teams [22].

the total average reduction rate is about 0.2689 which is very colse to $\sigma = \frac{1}{4}$ while the scale factor is about 3.7184. So the complexity is bounded by $\frac{2\sigma}{1-2\sigma} Cnnz\,(A) = 1.1636 Cnnz\,(A)$. Considering a scaling rule for this reduction scheme with $400 \times (3.7184)^{10} \approx 202125475$, it can reduce a power grid of about 200 Million nodes easily to 400 nodes within 11 levels. Thus, the double pairwise aggregation technique reduces the memory consumption rapidly which ensures a linear complexity in memory usage.

5.4 PowerRush on TAU Contest

The first annual Power Grid Simulation Contest has been successfully organized in TAU workshop by IBM Austin Research Lab to boost the academic research in power grid verification [22]. **PowerRush** is also evaluated with other nine simulators on the same HW/SW platforms by a set of real life industrial benchmarks in this contest [23]. The runtime and memory chart of top three teams is shown in Figure 5. Among them, the other two simulators **SEVA** and **TicTac** are also illustrated in [24][25]. Comparing with the other teams, **PowerRush** obtained a best convergence and extremely low memory consumption. As shown in Table 3, **PowerRush** uses only about **40%** runtime and **75%** memory respectively compared with the second place performance of other simulators. Excitedly **PowerRush** is expected to explore a widely practical use in industrial world.

Table 3: Results on all 11 benchmarks for top 3 teams.

Team Name	CPU Score	Memory Score	Error Score
PowerRush	136	260	0
SEVA	397	330	0
TicTac	242	634	0

6. CONCLUSIONS

We have presented the detail implementation of **PowerRush**. Aggregation based algebraic multigrid with K-Cycle accelerating is adopted as an implicit preconditioner for CG iteration method to solve the power grid. The double pairwise aggregation scheme uses two passes of a pairwise matching algorithm to ensure low setup cost both in runtime and memory usage. The K-Cycle scheme is a recursively accelerated W-Cycle where acceleration is performed by finding the optimal linear combination of two iterations. Thus a linear complexity solver is realized fully with algebraic information in the system matrix only. Also by taking the advantage of effective SPICE Parser and robust circuit Builder, **PowerRush** has shown to be an effective and robust simulator with a linear complexity for real power grid analysis.

7. ACKNOWLEDGEMENTS

The authors would like to thank Zhuo Li for his careful organizing effort on the contest, and to thank Sani R. Nassif for introducing this contest for us.

8. REFERENCES

[1] Tsung Hao Chen and Charlie Chung-Ping Chen. Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods. In *Proceedings of DAC*, pages 559–562, 2001.

[2] Min Zhao, Rajendran V. Panda, Sachin S. Sapatnekar, Tim Edwards, Rajat Chaudhry, and David Blaauw.

Table 4: Grid reduction efficiency of Double Pairwise Aggregation scheme

Benchmark	Initial Size	Reduction Level									Reduction Rate*
		1	2	3	4	5	6	7	8	9	
ibmpg3/GND†	151148	42535	11817	3250	880	239	-	-	-	-	0.2753
ibmpg4/GND†	303712	78876	21527	5822	1552	407	105	-	-	-	0.2650
ibmpg5/GND†	291559	75523	20095	5320	1412	377	-	-	-	-	0.2645
ibmpg6/GND†	430586	114936	31314	8657	2421	663	174	-	-	-	0.2720
ibmpgnew/GND†	272655	75804	21113	5804	1574	416	108	-	-	-	0.2711
thupg1	3261850	1016250	302693	82384	21244	5404	1369	347	-	-	0.2715
thupg2	6014504	1876044	559178	152303	39264	9973	2524	640	163	-	0.2694
thupg3	7856200	2448528	729549	198959	51293	13032	3310	838	213	-	0.2694
thupg4	10259771	3198126	952769	259647	66915	16990	4303	1089	275	-	0.2691
thupg5	13627506	4246482	1265492	344802	88799	22548	5720	1445	366	-	0.2691
thupg6	17216192	5365326	1598894	435519	112230	28500	7207	1824	460	117	0.2674
thupg7	18690184	5824651	1735814	473105	121978	30992	7852	1985	502	128	0.2676
thupg8	27661623	8620564	2568323	700007	180413	45840	11607	2941	744	188	0.2674
thupg9	31518068	9817025	2925390	797496	205484	52169	13205	3342	847	215	0.2675
thupg10	38415188	11967133	3566144	972293	250721	63664	16109	4074	1030	264	0.2677

† GND means that only the single net of ground network is used.
* Average reduction rate of all reduction levels

Hierarchical analysis of power distribution networks. In *Proceedings of DAC*, pages 150–155, 2000.

[3] Haifeng Qian, Sani R. Nassif, and Sachin S. Sapatnekar. Power grid analysis using random walks. *ACM Transactions on CAD*, 24(8):1204–1224, 2005.

[4] Kai Sun, Quming Zhou, Kartik Mohanram, and Danny C. Sorensen. Parallel domain decomposition for simulation of large-scale power grids. In *Proceedings of ICCAD*, pages 54–59, 2007.

[5] Stephen Cauley, Venkataramanan Balakrishnan, and Cheng Kok Koh. A parallel direct solver for the simulation of large-scale powerground networks. *ACM Transactions on CAD*, 29(4):636–641, April 2010.

[6] J. M. S. Silva, Joel R. Phillips, and L. Miguel Silveira. Efficient simulation of power grids. *ACM Trans. on CAD*, 29(10):1523–1532, October 2010.

[7] Joseph N. Kozhaya, Sani R. Nassif, and Farid N. Najm. Fast power grid simulation. In *Proceedings of DAC*, pages 156–161, 2000.

[8] Joseph N. Kozhaya, Sani R. Nassif, and Farid N. Najm. A multigrid-like technique for power grid analysis. *ACM Transactions on CAD*, 21(10):1148–1160, 2002.

[9] H. Su, E. Acar, and S. R. Nassif. Power grid reduction based on algebraic multigrid principles. In *Proceedings of DAC*, pages 109–112, 2003.

[10] Cheng Zhuo, Jiang Hu, Min Zhao, and etc. Power grid analysis and optimization using algebraic multigrid. *ACM Trans. on CAD*, 27(4):738–751, 2008.

[11] Petr Vaněk. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56:179–196, 1996.

[12] Pei Yu Huang, Huang-Yu Chou, and Yu-Min Lee. An aggregation-based algebraic multigrid method for power grid analysis. In *Proceedings of ISQED*, pages 159–164, 2007.

[13] YADRIAN C. MURESAN and YVAN NOTAY. Analysis of aggregation-based multigrid. *SIAM J. SCI. COMPUT.*, 30(2):1082–1103, 2008.

[14] Yvan Notay. Aggregation-based algebraic multilevel preconditioning. *SIAM J. MATRIX ANAL. APPL.*, 27(4):998–1018, 2006.

[15] C. W. Oosterlee and T. Washio. On the use of multigrid as a preconditioner. In *Proceedings of Ninth International Conference on Domain Decomposition Methods*, pages 441–448, 1996.

[16] Yvan Notay and Panayot S. Vassilevski. Recursive krylov-based multigrid cycles. *Numer. Linear Algebra Appl.*, 0:0–20, 2006.

[17] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[18] S. R. Nassif. *IBM power grid benchmarks*. [Online], http://dropzone.tamu.edu/∼pli/PGBench.

[19] Willian L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial, Second Edition*. SIAM Press, 2000.

[20] YVAN NOTAY. An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37:123–146, 2010.

[21] CHOLMOD. Sparse cholesky factorization and modification package. http://www.cise.ufl.edu/research/sparse/cholmod.

[22] Zhuo Li. *TAU 2011 power grid analysis contest*. www.tauworkshop.com/PREVIOUS/tau_2011_contest.pdf.

[23] Zhuo Li, Raju Balasubramanian, Frank Liu, and Sani Nassif. *2011 TAU Power Grid Simulation Contest: Benchmark Suite and Results*. In *Proceedings of ICCAD*, pages 478–481, 2011.

[24] Chung-Han Chou, Nien-Yu Tsai, Hao Yu, Che-Rung Lee, Yiyu Shi, and Shih-Chieh Chang. *On the Preconditioner of Conjugate Gradient Method - A Power Grid Simulation Perspective*. In *Proceedings of ICCAD*, pages 494–497, 2011.

[25] Zhiyu Zeng, Tong Xu, Zhuo Feng, and Peng Li. *Fast Static Analysis of Power Grids: Algorithms and Implementations*. In *Proceedings of ICCAD*, pages 488–493, 2011.